View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Digital library of Brno University of Technology

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTELIGENTNÍ SYSTÉM PRO ODPOVÍDÁNÍ NA OTÁZKY

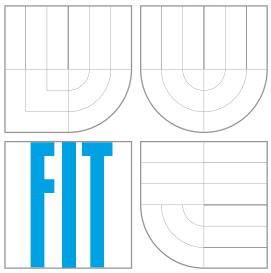
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB MIČULKA

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTELIGENTNÍ SYSTÉM PRO ODPOVÍDÁNÍ NA OTÁZKY

AN INTELLIGENT SYSTEM FOR QUESTION ANSWERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB MIČULKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUBOMÍR OTRUSINA

BRNO 2013

Abstrakt

Tato práce se zabývá problematikou zpracování otázek položených v přirozeném jazyce a jejich použitím ve vyhledávačích. Práce osvětluje základní principy funkčnosti vyhledávacích systémů s hlavním zaměřením na vyhledávání v databázích. Zásadní část článku pak tvoří popis návrhu a implementace programu *questionAnswering*, který slouží k vyhledávání dat v databázi projektu *ReResearch*. Čtenáři je vysvětlen postup při návrhu a tvorbě tohoto programu, popsány všechny zásadní problémy, jež se v průběhu této činnosti vyskytly, a nakonec jsou výsledky programu zhodnoceny pomocí standardních metrik.

Abstract

This work deals with problem about proccessing of natural language queries, which are asked in search engines. This work explains basic function principles of search engines, where the main focus is given to database search engines. The essential part of this article deals with system design and implementation of *questionAnswering*, which is used for searching information in the database of the *ReResearch* project. The reader is introduced to procedure of design and implementation of mentioned system and to the fundamental problems, that arose from this work. In the end, this system is evaluated with the standard metrics.

Klíčová slova

vědecký článek, vyhledávání, odpovídání na otázky, zpracování přirozeného jazyka, python

Keywords

scientific publication, search engine, question answering, natural language proccesing, python.

Citace

Jakub Mičulka: Inteligentní systém pro odpovídání na otázky, bakalářská práce, Brno, FIT VUT v Brně, 2013

Inteligentní systém pro odpovídání na otázky

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Lubomíra Otrusiny

.....
Jakub Mičulka
15. května 2013

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu Ing. Lubomíru Otrusinovi za cenné rady v průběhu tvorby této práce. Dále bych rád poděkoval Ing. Janu Kouřilovi za nasměrování v oblasti morfologických algoritmů a Aleši Kajzarovi za pomoc při tvorbě GUI.

© Jakub Mičulka, 2013.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

Pokud se podíváme na libovolný projekt většího rozsahu, svým charakterem alespoň technicky se dotýkající IT, jen málokterý z nich se v dnešní době obejde bez použití určitého druhu databáze. Jazyk SQL a jeho odvozeniny se staly standardem v oblasti zachovávání persistence dat a jejich následné správy.

Právě na problematiku operací s databází, konkrétně na vyhledávání požadovaných informací, je zaměřena i tato práce. Databáze, nad níž jsou vykonávány všechny dotazy popsané v tomto textu (bližší informace v kapitole 3) slouží jako úložiště dat pro projekt *ReResearch*, jenž je vypracováván pod záštitou *Skupiny znalostních technologií* při FIT VUT Brno. Cílem mé bakalářské práce je vypracovat vyhledávač pro tuto databázi, který bude jako vstup přijímat dotazy zapsané v přirozeném jazyku (angličtina) a bude tak pracovat jako inteligentní systém pro odpovídání na otázky, v případě potřeby však nebude problémem ani zadání dotazu v podobě základních klíčových slov. To by také mělo být hlavní výhodou oproti již existujícímu prototypu vyhledávače[1], kde je třeba pomocí přepínačů přesně specifikovat hledaný okruh témat. Vyhotovený projekt oproti tomuto řešení sám zjistí, jaký typ informací (publikace, projekt, původ autora...) uživatel požaduje a vrátí relevantní výsledek.

Jelikož lze vidět jasnou souvislost mezi vyhledávači pracujícími nad úzce zaměřenými databázemi a webovými vyhledávači, je tématice obecné funkčnosti vyhledávacích systémů věnována kapitola 2. V rámci této kapitoly je pak také popsána teorie z oblasti odpovídání na otázky, které se vypracovaný vyhledávač dotýká. Samotný návrh a implementace tohoto vyhledávacího systému, nazvaného *QuestionAnswering*, jsou pak čtenáři přiblíženy v kapitole 5, přičemž na některé techniky zpracování přirozeného jazyka, použité v programu, se zaměřuje už kapitola 4. Vyhodnocováním vytvořeného systému a výstupem použitých testů se následně zabývá sekce 6. A konečně, kapitola 7 přináší ucelený pohled na činnost systému při praktickém použití a nabízí srovnání s podobnými, již existujícími službami.

[1] http://athena2.fit.vutbr.cz:30101/rrsgui/www/

Kapitola 2

Inteligentní odpovídání na otázky

Pro dostatečné pochopení cílů tohoto projektu je hned zpočátku nutné seznámit čtenáře s ústředními pojmy, které jsou s daným tématem spjaty. Tato kapitola představuje adekvátní úvod do problematiky spojené s inteligentními systémy pro odpovídání na otázky.

2.1 Odpovídání na otázky

Pojmem *odpovídání na otázky* se v doslovné definici rozumí specializovaná odnož z odvětví vyhledávání informací (angl. *information retrieval*), kdy se systém pokouší nalézt odpovědi korespondující s otázkami, které jsou zadány v přirozeném jazyce [10].

Potřeba těchto systémů se s narůstajícím množstvím informací na internetu stále zvyšuje, jelikož současné vyhledávače umožňují v reakci na uživatelský dotaz vracet pouze seznamy dokumentů či odkazů ohodnocených dle relevance, avšak již neposkytují na dotaz přímou a jasnou odpověď. Právě na řešení tohoto problému se zaměřuje oblast *odpovídání na otázky*.

Z hlediska rozdílného přístupu k problémům v rámci jednotlivých kategorií lze pak celé toto odvětví rozčlenit (dle [6]) na níže podrobněji popsané části:

- cílové použití
- uživatelé
- typy dotazů
- typy odpovědí

Cílové použití Cílové použití na základě zdroje odpovědí lze rozdělit na tyto oblasti: strukturovaná data (databáze, případ popisovaného projektu), polostrukturovaná data (např. sloupce s komentáři v databázích) a volný text (text publikací apod.).

Dále lze rozlišovat mezi použitím systému v rámci neměnné množiny dokumentů (často testovací množiny), vyhledáváním na internetu nebo hledáním nad prostým textem. Z pohledu variability využití pak lze systémy pro odpovídání na otázky dělit na doménově nezávislé a doménově specifické (např. systémy určeny jako pomocné).

Uživatelé Cíloví uživatelé konečného systému mohou představovat jak absolutní začátečníky či příležitostné uživatele, tak experty z dané oblasti, kteří mohou např. systém používat ke své práci. Tyto okolnosti se pak projeví zejména na konečném uživatelském rozhraní systému, na rozdílných kladených dotazech a rovněž rozdílných očekávaných odpovědích.

Typy dotazů Dotazy lze dělit dle typu očekávaných odpovědí, mezi které patří věcné odpovědi, názory a shrnutí.

Podle samotné struktury dotazu lze pak rozlišovat dělení na dotazy očekávající jako odpověď ano/ne, tzv. "wh" dotazy (obsahující slova jako why, when, who... viz [6]), nepřímé požadavky (*I would like you to list...*) a příkazy (*List all authors from France*), přičemž poslední jmenované jsou použity i u vypracovaného systému *QuestionAnswering*.

Typy odpovědí Obdržená odpověď u systémů pro odpovídání na otázky může mít různou podobu v závislosti na zadaném dotazu a s ním souvisejícím očekávaném výstupu - pokud uživatel očekává podrobné vysvětlení, bude získaná odpověď logicky delší, než v případě, kdy je vznešen dotaz např. na jméno konkrétní osoby.

Při použití s množinou textů je pak také bráno v úvahu, zda je výsledná získaná odpověď tvořena extrakcí částí zdrojového textu nebo je vygenerována na základě obdržených informací (obvykle opět části různých vět z různých dokumentů), které jsou následně zpracovány pro účely dostatečně informativního výstupu [6].

Odpovědi systému *QuestionAnswering* tvoří data přímo získaná z databáze, jedná se tedy o zvláštní případ extrakce.

S popsaným tématem *odpovídání na otázky* přímo souvisí problematika vyhledávacích systémů, konkrétně pak webových vyhledávačů a dále systémů, hledajících data nad úzce zaměřenou databází. I když v této oblasti není pravidlem psát své dotazy v přirozeném jazyce a skutečně inteligentní systémy se stále objevují vzácně, je teoretická znalost těchto typů softwaru jedním z předpokladů pro úspěšné dosažení cílů tohoto projektu.

2.2 Webové vyhledávače

Webové vyhledávače jsou využívány jako software, jehož prostřednictvím je možno vyhledávat a procházet informace, nacházející se na síti internet. Tyto informace mohou představovat jak webové stránky, tak i různé druhy dokumentů či souborů obecně. Webové vyhledávače pracují se svou databází, která zpravidla pokrývá obsah velké části globální internetové sítě a v případě potřeby nabídnou uživateli co nejrelevantnější odpověď na jeho dotaz.

Mezi nejznámější webové vyhledávače patří *Google*[1], *Yahoo*[2], *Bing*[3], *Ask*[4] či u nás populární *Seznam*[5].

2.2.1 Princip funkčnosti webových vyhledávačů

Samotná činnost webových vyhledávačů se dá rozdělit na tři základní části:

- procházení (crawling)
- indexování (indexing)
- vyhledávání (searching)

[1] www.google.com/
[2] www.yahoo.com/
[3] www.bing.com/
[4] www.ask.com/
[5] www.seznam.cz/

V první fázi přichází na řadu tzv. *crawler*, někdy nazývaný též *spider*, který prochází internetovou síť a získává informace o obsahu jednotlivých webových stránek. Získaná data jsou poté procházena, zpracovávána (např. jsou extrahována klíčová slova z hlaviček stránek, nadpisů apod.) a ve vhodné formě uložena jako index do databáze.

Indexování se provádí za účelem pozdějšího, co nejrychlejšího, vyhledání informace požadované uživatelem - kdykoliv uživatel zadá svůj dotaz, je tomuto dotazu přiřazen určitý index, k němuž se následně v databázi hledá co nejpřesnější shoda. Úspěšnost vyhledávače pak závisí na relevanci vrácené množiny výsledků - i když existují miliony stránek obsahujících zadané slovo, některé jsou přece jen relevantnější, než jiné. Pro určování relevance a s tím souvisejícím zobrazením pořadí výsledků jsou pak používány specializované algoritmy [7].

2.3 Úzce zaměřené databázové vyhledávače

Narozdíl od webových vyhledávačů je tento druh vyhledávacích systémů zaměřen na práci nad databází, která obsahuje data týkající se konkrétní oblasti lidské činnosti. Nejčastějším případem takových datových úložišť jsou databáze pro skupiny působící v oblasti vědy a výzkumu. Tyto databáze pak obsahují např. informace o publikacích, jejich autorech a další doplňující údaje. Právě tomuto druhu informací, resp. databází a k nim korespondujícím vyhledávačům se bude věnovat tato podkapitola.

Mezi známé databázové vyhledávače patří mimo jiné *CiteSeerX*[6], *DBLP*[7] nebo *MS Academic Search*[8]. Systém *ReResearch*[9], pro něhož je tvořen vyhledávač, který je předmětem této práce, je typickým příkladem úzce zaměřeného databázového vyhledávače.

2.3.1 Princip funkčnosti databázových vyhledávačů

Zásadní problém, jenž bude řešen také v rámci implementace systému *QuestionAnswering*, zde představuje transformace uživatelských dotazů do jazyka databáze. Jelikož se nepředpokládá, že koncový uživatel oplývá znalostí jazyka SQL či jeho modifikací, je třeba umožnit mu zadat svůj požadavek prostřednictvím přirozeného jazyka, a to buď formou hesel, nebo rovnou celých vět.

Problematika převodu přirozeného jazyka na SQL dotazy je v rámci vědeckých prací již mnohokrát zpracována (např. [9], [3]), avšak z důvodu značného množství přístupů k tomuto problému se nedá doporučit či vyzdvihnout jeden konkrétní algoritmus. Můžeme však předpokládat, že na základní úrovni dekompozice se téměř vždy setkáme s následujícím postupem:

- morfologická analýza
- syntaktická analýza
- sémantická analýza
- tvorba dotazu

Ne vždy musí být přítomny všechny části tohoto výčtu (např. morfologická analýza může být brána jako součást syntaktické analýzy) nebo naopak může být celý proces převodu složitější, zmíněný seznam činností lze však považovat za rozumný použitelný základ.

[6] citeseerx.ist.psu.edu
[7] www.dblp.org
[8] academic.research.microsoft.com
[9] http://athena2.fit.vutbr.cz:30101/rrsgui/www/

Po úspěšném překladu do SQL jazyka pak už jen stačí položit tento dotaz databázi a zobrazit získané výsledky. Celý zde popsaný proces je samozřejmě složitější a více se mu budeme věnovat v kapitole 5.

2.4 Srovnání webových a databázových vyhledávačů

Zásadní rozdíl mezi oběma druhy popisovaných vyhledávačů spočívá v zaměření na typ dat, jež lze prostřednictvím vyhledávacího systému nalézt. Informace poskytované úzce zaměřenými vyhledávači jsou v ideálním případě přehledně organizovány a průběžně aktualizovány, což má za následek vysokou relevanci nalezeného obsahu. Z tohoto důvodu se však také často můžeme setkat s komerčními službami, které například umožní uživateli vyhledat požadovaná data, pro jejich detailnější zobrazení je však již třeba mít na dané službě účet a za její využívání platit adekvátní částky.

Naproti tomu internetové vyhledávače poskytují své služby zdarma a jsou volně přístupné jakémukoliv uživateli internetu, z pohledu relevance dat ale udržují natolik rozsáhlou databázi informací, díky čemuž nelze vždy čekat skutečně uspokojivý výstup. Celkově však internetové vyhledávače poskytují skutečně obecný přehled takřka ve všech myslitelných oblastech, a tak není problém dostat se i k typu informací, které systémy pracující nad databázemi ani zdaleka nepokrývají.

Kapitola 3

Databáze projektu ReResearch

Základním předpokladem pro úspěšné vypracování vyhledávače *QuestionAnswering* je podrobná znalost datového úložiště, nad nímž bude program operovat. Před samotným procesem návrhu programu tak bylo nutno provést studii databáze projektu *ReResearch* (dále také jako *RRS*), což je zároveň hlavním předmětem této kapitoly.

3.1 Projekt *ReResearch*

Projekt *ReResearch* vzniká pod záštitou *Výzkumné skupiny znalostních technologií(KNOT)* při FIT VUT v Brně.

Cílem tohoto projektu je vybudování znalostního systému, jehož základem jsou vědecké články a publikace, které slouží jako hlavní zdroj informací o dění na akademické půdě. Principem funkce celého systému je sběr dat o publikacích, autorech, organizacích, projektech atp., budování znalostí nad získanými daty a interaktivní prezentace informací uživateli [5].

3.2 Databáze projektu *ReResearch*

Pro upřesnění pojmu "databáze projektu RRS" je třeba konstatovat, že vždy, když bude o této databázi řeč, půjde o datovou databázi systému. Kromě ní se v projektu nachází ještě databáze řídícího systému, její popis však není předmětem této práce. Samotná datová databáze je zprovozněna pod systémem *PostgreSQL*.

3.2.1 Databázový systém PostgreSQL

PostgreSQL[1] představuje plnohodnotný databázový systém s otevřeným zdrojovým kódem. Primárně je vyvíjen pro *Linux*, resp. pro *unixové systémy*, funguje však i pod *OS Windows*. *PostgreSQL* stoprocentně splňuje podmínky *ACID*[2] a uživateli dokáže nabídnout plnou podporu cizích klíčů, operací JOIN, pohledů, triggerů a uložených procedur. Nezanedbatelnou výhodou tohoto databázového systému je pak jeho šíření pod BSD[3] licencí, která umožňuje neomezené bezplatné používání, distribuci a modifikaci, a to jak pro nekomerční, tak i komerční využití.

[1] http://www.postgresql.org/about/
[2] Atomic, Consistent, Isolated, Durable
[3] Berkeley Software Distribution

3.2.2 Obsah databáze RRS

Samotná databáze RRS se dá logicky rozdělit na tyto typy tabulek [4]:

- tabulky představující entitní množiny
- spojovací tabulky pro vyjádření N:N vztahu mezi entitními množinami
- tabulky typů (číselníky)
- metatabulky (metadata)

První otázkou, která byla nastolena na začátku vývoje vyhledávače, bylo, nad jakou databází a s jakými tabulkami program vůbec bude pracovat. Jelikož se předpokládalo, že vyhledávání bude zaměřeno zejména na základní údaje, jako publikace autorů nebo jejich geografická poloha, bylo by zbytečné určitým způsobem využívat tabulky typů či metatabulky. Z toho tedy vyplývá, že pro další práci postačují tabulky základních entitních množin a jejich spojovací články.

Po tomto rozhodnutí bylo ještě nutno specifikovat server a na něm běžící databázi, v níž bude vyhledávání prováděno. Na výběr bylo ze tří datových úložišť - jakožto nativní úložiště využíval systém RRS databázi *data*, nacházející se na serveru Athena3. Od této primární databáze se dalo očekávat, že se zde budou vyskytovat data v nejlepší kvalitě, a tedy i data nejvíce vyhovující požadavkům vytvářeného programu, avšak tento předpoklad nebyl potvrzen. Základním problémem zde byla chybějící data ve spojovacích tabulkách *j_loca_pers* a *j_loca_keyw*, pro požadované účely se tak jednalo o nevyhovující databázi.

Druhá dostupná databáze, DB *test* na serveru Athena3, využívána jako základní testovací schéma pro pokusy, testování a vyhodnocení, již nabídla užitečnější informace co do počtu položek v databázi, avšak v konečném důsledku toto úložiště použito nebylo, neboť kvalitnější alternativu poskytlo databázové schéma uložené na serveru nlpmosaic. Toto schéma, nazváno *test*, představuje databázi s největším počtem položek u všech potřebných tabulek. I když se jedná o výsledek automatické extrakce dat z textů, při zběžné kontrole dat z nativní DB a porovnání s daty ze serveru nlpmosaic nebyly nalezeny výrazné rozdíly, které by bránily uspokojivému provozu koncového projektu, a tak bylo rozhodnuto právě o využití této DB. Pro porovnání velikostí všech zkoumaných databází je možno nahlédnou do tabulky 3.1, která ukazuje počty položek v jednotlivých datových úložištích.

•	**athena3**		**nlpmosaic**
tabulka DB	*test*	*data*	*test*
publication	231731	1505416	1097786
person	896575	319012	1007653
project	22644	0	22543
organization	42068	843	54913
keyword	11551	1008	57440
j_pers_publ	502019	3667686	2419731
j_loca_pers	60690	0	60690
j_publ_keyw	12989	986	88822
j_proj_keyw	62283	0	62283

Tabulka 3.1: *Počet záznamů u jednotlivých tabulek v uvedených DB na serverech*

3.2.3 Dotazy vytvořené na základě stavu databáze RRS

V tabulce 3.1 jsou uvedeny pouze ty databázové tabulky, které obsahují dostatečné množství záznamů či k nim lze nalézt potřebné spojovací články. Už na základě této analýzy se dalo určit, jaké otázky bude zhruba možno pokládat a podle toho navrhnout strukturu dotazů.

Obecně se dá říci, že největším problémem, co se týče naplněnosti entitních množin záznamy, je absence spojovacích tabulek. Konkrétně lze tento fakt ilustrovat např. na tabulce *event*, jež na serveru *nlpmosaic* obsahuje celkem 35542 záznamů, avšak z důvodu velice malého množství položek ve spojovacích tabulkách j_even_<druhá tabulka> je tato množina pro mé účely nepoužitelná.

Ani uspokojivá naplněnost tabulek však ještě neznamená, že je vše v pořádku a že vyhledávání nebude činit žádné problémy. Později jsem totiž narazil na druhý, neméně palčivý problém - tím byl neadekvátní stav záznamů v tabulkách. Této nepříjemnosti se však budu věnovat až v dalších kapitolách, neboť přímo souvisí s úspěšností mého vyhledávače, a tím i s výsledky testů.

Následuje seznam některých dotazů, které byly navrženy před samotným začátkem práce na programu, přičemž tyto dotazy jsou v konečné verzi pokládány databázi.

Vyhledání informací o zadané osobě
Příklad dotazu: *Give me information about David Johnson*
SQL dotaz:

```
SELECT pers.full_name, publ.title, proj.title, loc.country, cont.email
FROM data_09_test.person pers
LEFT JOIN data_09_test.j_loca_pers jLP ON jLP.person_id = pers.id
LEFT JOIN data_09_test.location loc ON jLP.location_id = loc.id
LEFT JOIN data_09_test.j_pers_proj_works jPPW ON jPPW.person_id = pers.id
LEFT JOIN data_09_test.project proj ON jPPW.project_id = proj.id
LEFT JOIN data_09_test.j_cont_pers jCP ON jCP.person_id = pers.id
LEFT JOIN data_09_test.contact cont ON jCP.contact_id = cont.id
LEFT JOIN data_09_test.j_pers_publ jPP ON jPP.person_id = pers.id
LEFT JOIN data_09_test.publication publ ON jPP.publication_id = publ.id
WHERE pers.full_name ~* 'David␣Johnson';
```

Údaje o autorech pocházejících z dané země
Příklad dotazu: *List authors from France and their publications*
SQL dotaz:

```
SELECT pers.full_name, publ.title, loc.country
FROM data_09_test.person pers
JOIN data_09_test.j_loca_pers jLocaPers ON jLocaPers.person_id = pers.id
JOIN data_09_test.location loc ON jLocaPers.location_id = loc.id
JOIN data_09_test.j_pers_publ jPersPubl ON jPersPubl.person_id = pers.id
JOIN data_09_test.publication publ ON jPersPubl.publication_id = publ.id
WHERE loc.country ~* 'Franc';
```

Poznámka: podobnou strukturu SQL kódu pak má i dotaz na autory z dané univerzity, pouze jsou změněny názvy hlavních a spojovacích tabulek

Vyhledání kontaktu
Příklad dotazu: *Give me contact on A. Abidi*
SQL dotaz:

```
SELECT DISTINCT pers.full_name, cont.email
FROM data_09_test.person pers
JOIN data_09_test.j_cont_pers jCP ON jCP.person_id = pers.id
JOIN data_09_test.contact cont ON jCP.contact_id = cont.id
WHERE pers.full_name ~* 'a. abidi';
```

Zapojení tabulky *keyword*
Entitní množina *keyword* je používána hned u několika typů uživatelských dotazů - jednak u těch zaměřených na projekty a jednak u zadání jednotlivých pojmů. Následujících SQL dotazů připadajících na jednu otázku je vždy více, neboť se takovým způsobem zajistí pokrytí co největšího spektra informací, které lze na daný dotaz nalézt. V praxi jsou tyto SQL dotazy kladeny na databázi vícekrát a smyslem tohoto snažení je se co nejvíce přiblížit informacím získaných z již zavedených vyhledávačů

Příklad dotazu: *Give me information about robotics*
SQL dotazy:

```
SELECT pers.full_name, proj.title
FROM data_09_test.person pers
RIGHT JOIN data_09_test.j_pers_proj_works jPPP ON jPPP.person_id = pers.id
RIGHT JOIN data_09_test.project proj ON jPPP.project_id = proj.id
WHERE proj.title ~* 'robot';

SELECT pers.full_name, proj.title
FROM data_09_test.person pers
RIGHT JOIN data_09_test.j_pers_proj_works jPPP ON jPPP.person_id = pers.id
RIGHT JOIN data_09_test.project proj ON jPPP.project_id = proj.id
RIGHT JOIN data_09_test.j_proj_keyw jPK ON jPK.project_id = proj.id
RIGHT JOIN data_09_test.keyword keyw ON jPK.keyword_id = keyw.id
WHERE keyw.title ~* 'robot';

SELECT pers.full_name, publ.title
FROM data_09_test.person pers
RIGHT JOIN data_09_test.j_pers_proj_works jPPP ON jPPP.person_id = pers.id
RIGHT JOIN data_09_test.project proj ON jPPP.project_id = proj.id
RIGHT JOIN data_09_test.j_pers_publ jPP ON jPP.person_id = pers.id
RIGHT JOIN data_09_test.publication publ ON jPP.publication_id = publ.id
RIGHT JOIN data_09_test.j_publ_keyw jPK ON jPK.publication_id = publ.id
RIGHT JOIN data_09_test.keyword keyw ON jPK.keyword_id = keyw.id
WHERE keyw.title ~* 'robot';
```

Výše zmíněné příklady nelze brát jako kompletní seznam všech dotazů, které jsou v projektu vytvářeny. Kromě uvedených kódů dále vyhledávač pracuje s jejich kombinacemi (*najdi finské autory z univerzity Brno*) či jednoduššími verzemi (*vypiš seznam francouzských autorů bez jejich publikací*). Dále je podporováno i zadávání roků (*vypiš publikace francouzských autorů z roku 2003*) a číselné omezení dotazů, co do počtu konečných vypsaných položek. Kapitolou samou pro sebe pak jsou dotazy pracující se seznamem nejcitovanějších autorů, kdy jsou do klauzule WHERE přidány jména z tohoto seznamu. V případě, že uživatel zadá pouze dotaz na nejcitovanější autory, není ani potřeba vytvářet SQL kód a uživateli je zobrazen přímo obsah daného seznamu.

3.2.4 Otázky na citovanost autorů

Jak je patrné výše, program umí odpovídat i na otázky, týkající se citovanosti jednotlivých autorů. Praktickému provedení však na začátku bránil nedostatek potřebných dat v databázi - i když samotná tabulka *citations* obsahovala velice slušné množství záznamů, z důvodu zcela chybějící vazby na množinu persons nebylo možné jakékoliv údaje o citovanosti zjistit.

Z toho důvodu bylo rozhodnuto, že se pokusím najít seznam nejcitovanějších autorů na některém ze známých vědeckých portálů a případně takový seznam do projektu zapracuji. Následuje tedy výčet internetových stránek, které jsem v rámci hledání vhodných informací navštívil. Ke každému portálu je navíc uveden krátký popis s ohledem k řešenému problému.

Altmetric

Altmetric[4] představuje nástroje potřebné pro sledování úspěšnosti vědeckých článků, a to v podobě moderně působící webové aplikace. Data z této aplikace zahrnují i citovanost autorů, problémem je však nutnost si za tento obsah platit. Kromě toho však Altmetric umožňuje použít i volně dostupné webové API, které vrací data ve formátu JSON, a to i žebříčky citovanosti. Důvodem, proč tyto seznamy v projektu nevyužít, bylo, že žebříčky jsou až příliš konkrétní (nejcitovanější autoři za poslední týden, měsíc apod.) a celkový seznam nejcitovanějších autorů vůbec zde chybí.

Article level metrics

Z tohoto serveru[5] se dají stahovat měsíčně aktualizované CSV soubory, v nichž se nachází informace o citaci publikací na různých sítích (Scopus, Pubmed, Facebook, Twitter...). Tyto údaje se však netýkají autorů, ale přímo samotných publikací. Pro vypracování citovanosti autorů z těchto dat by tak muselo být použito mnoho dalšího materiálu, což v projektu nebylo žádoucí.

CitedIn

Tato služba[6] nabízí webové API pro vyhledávání tzv. *Pubmed identifiers*, což jsou odkazy do databáze, jež obsahuje bibliografické citace a abstrakty článků. I přes nezpochybnitelnou užitečnost takového vyhledávače jsem zde však nenalezl žádný ucelený seznam, který bych ve svém projektu mohl využít.

CiteSeerX

Portál CiteSeerX[7] se zpočátku jevil jako nepříliš zajímavý - žádné API a nejednoznačný způsob přístupu k jejich datům příliš dobrý první dojem neudělal. Poté, co jsem však objevil seznam[8] citovanosti všech autorů, obsahující 10000 položek, navíc v podobě pro parsing vyhovujícího HTML kódu, zařadil jsem tento portál mezi uvažovaná řešení pro využití v projektu.

Nic efektivnějšího již později nalezeno nebylo, a tak se seznam citovaných autorů stal podkladem, z něhož nyní vyhledávač čerpá. Tento seznam se nachází ve složce spolu se

[4] http://www.altmetric.com/index.php
[5] http://article-level-metrics.plos.org/
[6] http://citedin.org/
[7] http://citeseerx.ist.psu.edu/
[8] citeseerx.ist.psu.edu/stats/authors?all=true

zdrojovými soubory projektu a nese název `authorCitesList.pickle`. Jedná se o soubor, vytvořený pomocí modulu `pickle` a jeho obsahem je *list* (implementace seznamu v jazyce *Python*), obsahující další tříprvkové *listy* s touto strukturou:

```
[pořadí autora, jméno autora, počet citací autora]
```

Google Scholar a MS Academic Search

Oba[9] tyto "konkurenční" portály jsou zařazeny do společné podkapitoly z důvodu, že jejich filosofie je velmi podobná. Na každém z těchto serverů si lze nechat zobrazit stránky autorů a na nich články spolu s jejich citovaností. Souhrnný seznam nejcitovanějších autorů na obou serverech chybí, avšak podrobné informace o citovanosti právě u autorských profilů byly potencionálním impulsem k jejich zpracování a následném vyhodnocení. Z hlediska relevance získaného seznamu by se jednalo o bezkonkurenční řešení, vzhledem k netriviálnosti parsingu údajů byl však tento postup nakonec zamítnut.

ImpactStory

Jde o server[10], který výzkumníkům umožňuje sledovat úspěšnost či vliv své publikace prostřednictvím sběru dat z různých zdrojů. Tyto zdroje zahrnují např. počty sdílení na sociálních sítích či citovanost článku. Pro mé účely se ale na serveru nenacházelo nic zajímavého.

Scopus

Scopus[11], jakožto jedno z největších datových úložišť abstraktů článků a citací, se při běžném používání webového rozhraní ukázal jako nepříliš nápomocný. K řešení mého problému se teoreticky daly použít soubory zasílané e-mailem na požádání, avšak tyto soubory obsahovaly pouze informace ohledně publikací na zadané téma a navíc pokrývaly značné množství údajů, jež pro mě byly nezajímavé.

Je důležité říci, že v této podkapitole jsou zmíněny pouze některé servery, které byly v rámci průzkumu případného použití analyzovány. Prozkoumaných portálů bylo ve skutečnosti více (mimo jiné např. DBLP, Mendeley, Scirus...), avšak ani jeden z nich pro odpovídající účely nenabídl uspokojivé minimum informací, vhodných k řešení daného problému nebo ke zmínění v této podkapitole.

3.3 Použité prostředky pro komunikaci s databází

Jelikož pro samotnou implementaci vyhledávače byl zvolen jazyk Python, nezbytnou součástí pro přístup k DB skrz tento jazyk představovala knihovna *PsycopPg2*. Za zmínku pak stojí program *phpPgAdmin*, jenž nabídl přehledné GUI a posloužil tak k pohodlnější analýze databáze.

[9] http://scholar.google.cz/, http://academic.research.microsoft.com/
[10] http//:www.impactstory.org
[11] http://www.scopus.com/

3.3.1 Knihovna *PsycoPg*

PsycoPg, který jsem používal ve verzi *2.7*, je modul pro Python, jenž nabízí uživateli kompletní implementaci *Python DB API 2.0*[12] a je používán pro přístup k *PostgreSQL* systémům z programovacího prostředí. Tento modul nabízí komunikaci s DB jak ze strany serveru, tak ze strany klienta a je vyvinut pro použití ve vícevláknových aplikacích [13]. Ve vypracovaném projektu byla tato knihovna použita jako prostředník mezi samotným programem a cílovou databází.

3.3.2 *PhpPgAdmin*

PhpPgAdmin je webový administrační nástroj pro přístup k *PostgreSQL* databázím. Kromě přehledného výpisu tabulek DB a dat v ní uložených je možno provádět všechny myslitelné databázové operace jako přidávání a mazání záznamů, vytváření tabulek či triggerů a další související úkony [14]. Tento nástroj byl využit zejména při analýze obsahu RRS databáze.

[12] Python Database API Specification v2.0
[13] Zdroj: http://initd.org/psycopg/docs/
[14] Zdroj: http://phppgadmin.sourceforge.net/doku.php

Kapitola 4

Metody zpracování přirozeného jazyka využité v programu

Nezbytnou součástí systému *QuestionAnswering* jsou také některé metody zpracování přirozeného jazyka. Konkrétně se jedná o využití *Named Entity Recognition* systému (česky "systému pro rozpoznávání pojmenovaných entit") a metody *stemmingu*. V této kapitole bude popsán jejich teoretický princip a následně i praktické využití v projektu.

4.1 NER systém

Named entity recognition, dále pod názvem *NER*, označuje proces, spadající do oblasti *vyhledávání informací*. Tento proces představuje kombinovanou úlohu, kdy je nejprve nalezeno slovo či slovní spojení, představující určitý druh slovního pojmenování a následně je tomuto spojení přiřazeno označení identifikující jeho kategorii [8].

Současné *NER* systémy se zaměřují zejména na detekci jmen lidí, názvů míst nebo organizací. Specializované systémy pak počítají s rozpoznáváním např. komerčních produktů, zbraní nebo také uměleckých děl a dalších kategorií. Společným jmenovatelem pro všechny tyto systémy je zaměření na určité typy hesel (neboli slovních spojení, vyskytujících se v daném jazyce či oblasti) a předem daná množina kategorií, do nichž lze nalezená hesla řadit. Tabulka 4.1 (převzata z [8]) zobrazuje některé často pokrývané kategorie z oblasti *NER*, vzhledem ke značné šíři tohoto tématu je však třeba brát ji čistě jako ilustrační a počítat i s existencí dalších kategorií, které již v jejím rámci nejsou popsány.

Kategorie	**Příklady entit zapadajících do kategorie**
Osoby	Jednotlivci, fiktivní charaktery, malé skupiny lidí
Organizace	Společnosti, politické strany, náboženské skupiny, univerzity
Lokace	Státy, provincie, města
Zařízení	Mosty, budovy, letiště
Dopravní prostředky	Automobily, letadla a vlaky

Tabulka 4.1: *Některé kategorie NER systémů a k nim příslušející entity*

4.1.1 Přehled algoritmů využívaných pro tvorbu NER

Základním požadavkem kladeným na *NER* systémy je co nejefektivnější schopnost rozpoznat jednotlivé kategorie v rámci množiny neznámých slov, přičemž míra této schopnosti je přímo závislá na použitých pravidlech z oblasti *rozpoznávání a klasifikace* [12].

Zatímco první *NER* systémy využívaly ručně vytvářených klasifikačních pravidel, v současné době jsou pro účely trénování využívány techniky strojového učení, přičemž mezi metody zahrnuté do této oblasti patří např. *skryté Markovovy modely* (v souvislosti s NER popsány v [2]) nebo metoda *CRF* [11]. Zatímco tyto dvě metody ke své činnosti jako prerekvizitu vyžadují korpus anotovaných dat (jedná se o *techniku učení s učitelem*), metoda *shlukování*, jejíž varianta pro *NER* je přiblížena např. v [1], je této nutnosti zbavena a poskytuje tak dobrý předpoklad pro použití v oblastech, jež existující korpusy nepokrývají. Pro podobné účely lze využít i tzv. *bootstrapingu*, konkrétně z této techniky vzešlou metodu *mutual bootstraping*, která byla prvně představena v roce 1999 v [17]. Metody založené na *bootstrapingu* vyžadují jako prerekvizitu pouze malou množinu klíčových slov, z nichž jsou následně odvozovány další důležitá hesla.

4.1.2 Stanford NER

NER systém ve vypracovaném vyhledávači zastupuje *Stanford NER*[1]. *Stanford NER* je napsaný v jazyce Java a dává uživateli na výběr z několika připravených modelových tříd vytvořených za pomoci metody *CRF*[2], z nichž každá obsahuje odlišné množství kategorií. Pro účely výsledného programu je využíváno velikost písma nerozlišující třídy, která obsahuje čtyři kategorie - lokace, osoby, organizace a různé, což je pro plánované použití zcela dostačující.

4.1.3 Praktické využití v programu

Stanford NER tvoří součást třídy `qaFilter`, kde je v metodě `categorize` využíván jako analyzér přijatých slov, která jsou poté dle jejich přiřazené kategorie adekvátně zapracovány do klauzule WHERE v SQL dotazu.

Pro plnou funkčnost *Stanford NER* v jazyce Python je nutné použít *Python interface for Stanford NER*[3] a poté spustit java server, obsažený v distribuci *Stanford NER*. Samotná metoda `categorize` následně využívá tohoto systému pro určení kategorií *person*, *location* a *organization*, přičemž pokud není slovo rozpoznáno, je dle regulárních výrazů přijaté slovo označeno jako *year* nebo *project*.

4.2 Stemming

Stemming (česky "vytvoření základního tvaru") je v oblasti *vyhledávání informací* často využívaná metoda používaná k redukci slova na jeho základní tvar či kořen, a to za účelem zvýšení hodnoty *pokrytí* (viz kapitola 6.1) a získání většího počtu relevantních výsledků [18]. Jde o způsob, jak u různých morfologických variant jednotlivých slov docílit toho, aby byla vyhodnocena jako totožná [8]. Jako příklad lze uvést anglická slova *process*, *processing* a *processed*, kdy po aplikaci stemmovacího algoritmu dostaneme tvar *process*. Hlavní výhodou

[1] http://nlp.stanford.edu/software/CRF-NER.shtml
[2] Conditional Random Fields
[3] https://github.com/dat/pyner

stemmingu je, že umožňuje při zadání konkrétního dotazu nalézt výskyty daného slova ve všech morfologických tvarech, čehož je využíváno právě ve vyhledávačích.

4.2.1 Přehled základních stemmovacích algoritmů

Všechny stemmovací algoritmy lze na základě využívaného přístupu rozdělit do dvou skupin (dle [18]): přístup na základě pravidel a statistický přístup

Přístup na základě pravidel V rámci tohoto přístupu je vždy specifikována určitá množina pravidel, na základě níž je poté prováděn samotný *stemming*. V rámci těchto pravidel je mimo jiné uveden i seznam validních tvarů získaných slov a zároveň je nutné definovat speciální pravidla pro výjimečné situace.

Výhodou takového řešení je jeho rychlost v praktickém využití a v případě nasazení s anglickým jazykem také velmi vysoká míra uspokojivých výsledků. Mezi nevýhody naopak patří nemožnost zasazovat zpracovávaná slova do kontextů, nutnost jazykové expertízy k vyhotovení algoritmů a také relativně velký objem uchovávané množiny pravidel [18].

Z algoritmů jsou v souvislosti s tímto přístupem zmiňovány *Lovins stemmer* a v projektu využitý *Porter stemmer* (viz 4.2.2).

Statistický přístup Dle některých provedených studií ([18]) představují statistické stemmery kvalitní alternativu k výše popsaným stemmerům pracujících na základě pravidel. Jejich výhoda spočívá ve faktu, že nepotřebují k úspěšné implementaci jazykovou expertízu a místo toho využívají k učení rozsáhlé korpusy dat v konkrétním jazyce. Díky této skutečnosti dosahují nejlepších výsledků při nasazení s morfologicky komplexnějšími jazyky (francouzština, portugalština...), avšak v reálném využití je tato výhoda znevážena celkovou pomalostí statistických stemmerů.

K využívaným stemmovacím algoritmům této kategorie patří např. *Yet another suffix stripper (YASS)* nebo *Graph based stemmer (GRAS)* (oba blíže popsány v [18]).

4.2.2 Porter stemmer

Porter stemmer představuje algoritmus, který je určen pro odstraňování často se vyskytujících nebo často skloňovaných zakončení slov v anglickém jazyce. Je používán jako součást normalizace pojmů v oblasti *vyhledávání informací* [14], kde jde o jeden z nejvyužívanějších algoritmů. Pro účely tohoto projektu byl zvolen zejména z důvodu jeho rychlosti, rozšířenosti a jednoduchosti použití.

Porter stemmer vychází z původního *stemming* algoritmu, uveřejněného v [16] a vznikl z jednoho hlavního důvodu - přibývalo příliš mnoho verzí tohoto původního algoritmu, vzájemně se lišících svou funkcionalitou a bylo tedy třeba vytvořit normalizovanou a definitivní implementaci, vhodnou pro další šíření.

4.2.3 Praktické využití v programu

Jelikož bylo potřeba využít *Porter stemmer* napsaný pro jazyk Python, nejjednodušší cestu představuje použití implementace tohoto algoritmu jakožto součást knihovny NLTK[4].

[4] http://nltk.org/api/nltk.stem.html?highlight=porter%20stemmer#stem-package

Ve vypracovaném vyhledávači lze *Porter stemmer* nalézt ve třídě `qaFilter` a v metodě `filterWord`, kdy je na přijaté slovo aplikován algoritmus a takto získaný tvar vrácen hlavní větvi programu, kde je vložen do klauzule WHERE jako tvar slova, dle něhož se vyhledává.

Kapitola 5

Návrh a implementace programu

V momentě, kdy již byla známa struktura a obsah cílové databáze, bylo třeba začít se zabývat samotným principem funkčnosti programu a následně jeho implementací. Tato kapitola pojednává o všech nutných krocích, vedoucích ke konečné funkční aplikaci a o problémech, jež byly v průběhu práce nutné řešit.

5.1 Koncepční návrh programu

Už v původním ideovém návrhu se na základní úrovni dekompozice počítalo se třemi hlavními částmi - parserem, který přijme a zpracuje uživatelský vstup, dále s modulem pro samotnou tvorbu SQL dotazů a nakonec s programovou částí, starající se o zobrazení získaných výsledků přehlednou formou.

V konečném důsledku je v návrhu zahrnuto pět tříd, jejichž funkce jsou následující:

- *qaParser* - zpracování uživatelského vstupu
- *qaCountryProcess* - zpracování názvů zemí
- *qaSqlCreator* - řízení procesu tvorby SQL dotazu
- *qaFilter* - součást třídy *qaSqlCreator*, určuje tvar klauzule WHERE
- *qaPostProcess* - úprava získaných dat do uživatelsky přívětivé podoby

Konkrétněji je princip funkčnosti popsán a ilustrován v následujících podkapitolách.

5.1.1 Popis principu funkčnosti

Předpokládejme, že na začátku celého procesu stojí dotaz zadaný uživatelem. Podporovaným jazykem je angličtina, přičemž je možno svůj požadavek zadat jak ve formě klíčových slov, tak i v přirozeném jazyce.

Vstupní dotaz je poté přebrán parserem (modul `qaParser`), jehož úkolem je extrahovat z textu důležité údaje, jež budou použity v následujících částech programu. Výstup parseru tvoří dva seznamy - seznam klíčových slov a seznam doplňujících údajů (tím je myšlena např. specifikace konkrétní země či autora).

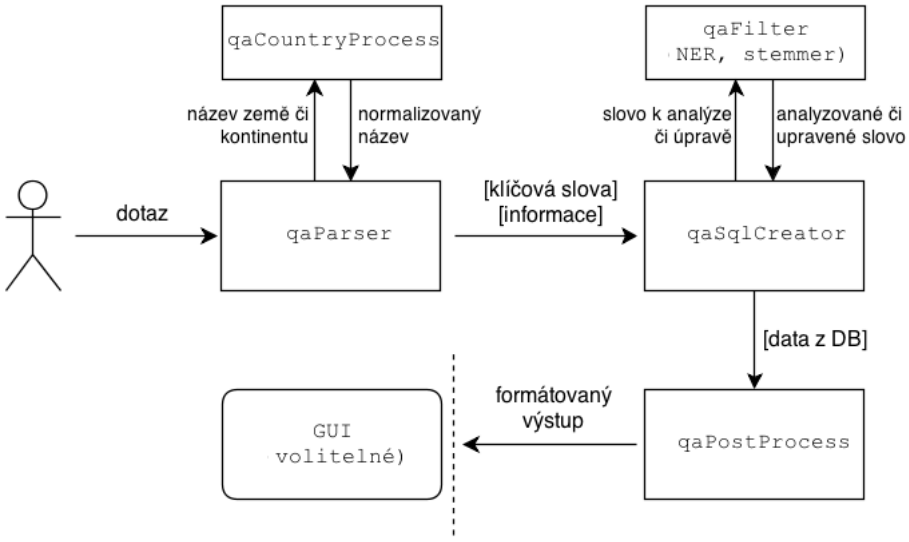
Jelikož se počítá i s dotazy, v nichž je název země či kontinentu zmíněn ve formě přídavného jména (czech, european authors) a další logické celky v architektuře návrhu již neprovádějí žádnou modifikací identifikovaných slov, je třeba zajistit, aby názvy zemí ve

výstupním listu parseru byly vždy totožné, ať už se jedná o přídavné či podstatné jméno. Tento problém řeší třída `qaCountryProcess`, jejímž primárním úkolem je konvertovat případné nalezené přídavné jméno do prvního pádu jména podstatného.

Výstup parseru přijímá třída `qaSqlCreator`, která představuje zřejmě nejdůležitější část vyhledávače - jejím úkolem je na základě obdržených seznamů vytvořit regulérní SQL dotaz, jenž následně bude položen databázi. Na tomto místě je třeba kontrolovat mnoho indikátorů, ovlivňujících finální podobu SQL kódu - např. zda bude konečný seznam omezen určitým počtem položek, zda je zapotřebí pracovat se seznamem nejcitovanějších autorů a zejména pak, jako jeden z nejdůležitějších dotazů, jak položit podmínku v klauzuli WHERE tak, aby bylo nalezeno co nejvíce relevantních položek. Konkrétněji je tato problematika rozebrána v podkapitole 5.2, pro obecné pochopení funkce programu je však postačující pouhý odkaz na další třídu s názvem `qaFilter`. Ta má (mimo jiné) na starosti právě vytváření odpovídajícího tvaru slova pro zmíněnou klauzuli a pokud je její činnost úspěšná, je cílový SQL dotaz kompletní a lze jej položit databázi.

Jakmile jsou z databáze získána požadovaná data, je zapotřebí zobrazit obdržené výsledky uživateli, a to prostřednictvím třídy `qaSqlCreator`. Protože lze vyhledávač použít jako *stand-alone* program, či jej využít ve spolupráci s webovým GUI, závisí funkce této třídy právě na zvoleném způsobu použití. V prvním uvedeném případě jsou získaná data v surovém stavu jen zkopírována na výstup (kde je převezme GUI a upraví do vyhovující podoby), v druhém případě je pak při použití speciálního vypínače dostupná možnost tisku dat do souboru/na výstup ve formátu přehledné ASCII tabulky. V tu chvíli je činnost programu u konce.

Pro základní pochopení funkčnosti by tento popis měl být dostačující, jako ilustrační obrázek je pak možno prohlédnou si diagram 5.1.



Obrázek 5.1: *Diagram popisující základní funkčnost programu*

5.2 Proces implementace programu

Základní řídící prvek ve vyhledávači představuje funkce `main()`, vyskytující se v souboru `qaMain.py`. Sekvence kódu, která je v této funkci obsažena, slouží jako řídící algoritmus

celého programu - jsou vytvářeny instance objektů a volány jejich metody. Navíc je odsud třeba kontrolovat, kolikrát bude daný dotaz položen databázi a dle tohoto faktu případně některé metody volat vícekrát.

5.2.1 Třída qaParser

Vstupním parametrem této třídy je řetězec přijatý od uživatele. Cílem je identifikovat v tomto řetězci důležitá slovní spojení a klíčová slova, která jsou pro potřeby vyhledávání důležitá, díky čemuž lze následně připravit podklady pro správnou funkci systému jako celku.

Jelikož zde základní předpoklad představuje vyhledávání nad tématicky omezenou množinou informací, je možno stejně tak počítat i s omezeným počtem klíčových slov, která budou k vyhledávání využita. Právě proto tvoří nejjednodušší cestu k identifikaci důležitých hesel *regulární výrazy*. Ty jsou zde rozděleny do několika kategorií:

- předložky
- číslovky
- národy a kontinenty
- klíčová slova
- pomocná slova

Pro určování jednotlivých typů slov je v metodě `parseAnswer` přítomen algoritmus, který se stará o všechny úkony s tímto problémem spojené. Vstupem zmíněného algoritmu je seznam jednotlivých slov zadaného řetězce s příznaky (na začátku nula) a jeho princip spočívá v následném procházení takto získaných prvků. V závislosti na zjištěném typu každého slova se pak provádí jedna z následujících akcí:

předložka: je volána metoda `prepositionProcess`, kde jsou dále vyhledávány důležité vazby na tuto předložku

číslovka: může se jednat o slovní či číselné vyjádření (počet slovního zadání číslovek omezen) - je volána metoda `keywordSearch`, v níž je vyhledáván podstatné jméno příslušející k dané číslovce

národnost nebo kontinent: myšleno jako přídavné jméno znamenající národnost (italian, english, french... konkrétní názvy států jsou zpracovány v rámci ostatních slov) - opět je volána metoda `keywordSearch` a hledáno související podstatné jméno. Mimo jiné je ale také nalezené slovo zkonvertováno přímo na název daného státu (italian -> Italy, french->France) s pomocí metody `normalizeNation`. Tato normalizace opět využívá regulárních výrazů a pochopitelně bude množina takto podporovaných slov vždy omezena (více v sekci 5.2.1).

klíčové slovo: ihned zařazeno do konečného seznamu klíčových slov

pomocné slovo: zde patří slova jako give, get, list, all... v případě jejich výskytu jsou zahozeny

Slova, která nevyhovují výše zmíněným kategoriím, jsou zařazeny do konečného seznamu doplňujících informací (v projektu název *info*), neboť se předpokládá, že jde např. o název autora či projektu. Zároveň je u každého zkontrolovaného slova změněn příznak z nuly na jedna.

Pomocné funkce třídy qaParser

Samotná hlavní metoda `parseAnswer` využívá několik pomocných funkcí, které se jí svým principem podobají.

Metoda `keywordSearch` slouží k vyhledávání podstatného jména náležícímu k danému slovu. Zpracováváme-li např. vstup "ten authors from Italy", pak je v hlavní smyčce programu při nalezení číslovky *ten* volána právě funkce `keywordSearch`, k této číslovce je vyhledáno podstatné jméno *authors* a celý řetězec pak uložen do výstupního seznamu keywords.

Druhá metoda `prepositionProcess` má za úkol hledat vazby na nalezené předložky *by, from, on, about, of, in*. Po nalezení předložky se volá právě tato metoda a probíhá cyklus velice podobný hlavní metodě `parseAnswer` - po nalezení slova, které k dané předložce náleží, je tato vazba uložena do výstupního seznamu info. Pro řetězec "ten authors from Italy" by metoda `prepositionProcess` uložila do seznamu vazbu "from Italy".

Třída qaCountryProcess

Jelikož je zpracovaný projekt velkou měrou založen na vyhledávání vazeb autorů na jednotlivé země, je v implementaci přítomna i třída, jež pokrývá práce týkající se zpracování názvů zemí. V této třídě se nachází metody `normalizeNation` a `contProccess`, které vrací k přijatému přídavnému jménu (italian, french, dutch...) normalizovaný název země (či zemí).

V případě první jmenované metody je v závislosti na přijatém přídavném jménu navráceno slovo, představující název země v prvním pádu. V současné době je podporováno deset nejčastěji se vyskytujících zemí v databázi, mezi které patří Belgie, Česká republika, Dánsko, Nizozemí, Anglie, Francie, Itálie, Portugalsko, Španělsko a Švédsko. Názvy těchto zemí lze tedy na výstup zadávat i ve formě přídavného jména a jednoduchý `if-else` proces se pak stará o jejich konverzi (jde o navrácení korespondujícího řetězce pomocí konstrukce `return`) do normalizovaného názvu. Speciální situaci zde představuje konverze přídavného jména *english*, kterému je přidělen normalizovaný název ve formátu *United Kingdom*. I když se taková transformace nemusí jevit jako ideální, je tato akce provedena s ohledem na velký výskyt názvu *United Kingdom* v databázi (jde o nejčastěji se vyskytující se lokaci v DB systému *ReResearch*) a převedení např. na slovo *England* by nedosahovalo požadovaných výsledků.

Metoda `contProccess` pak funguje na stejném principu jako `normalizeNation`, pouze je zde navracen seznam zemí, příslušejících k danému kontinentu. Podporovaná klíčová slova zahrnují zadání *european, african, asian a american*, a to i v prvním pádu.

5.2.2 Třída qaSqlCreator

Třída `qaSqlCreator` je zřejmě nejdůležitější částí tohoto projektu. Jejím hlavním účelem je vytvořit z přijatých seznamů SQL dotaz, který bude následně položen databázi. Zmíněné

seznamy tvoří výstup parseru, tedy seznam klíčových slov a seznam konkrétních upřesňujících hesel.

V první fázi je procházen seznam klíčových slov a na základě kontrolovaného řetězce je do slovníku s hlavními tabulkami (`mainTables`) přidán název korespondující entitní množiny (viz tabulka 5.1). Tato analýza počítá i s detekcí číslovek pro případné omezení konečného počtu vypsaných prvků a také s uvedením klíčových slov *most cited* jakožto indikátoru práce se seznamem nejcitovanějších autorů. Podobně je pak procházen i seznam `info`, kde je každý

Klíčová slova	**Přiřazený název tabulky v DB**
article(s), publication(s)	data_09_test.publication
author(s), person(s), researcher(s)	data_09_test.person
contact(s)	data_09_test.contact
project(s), research(es)	data_09_test.project
countr(y|ies)	data_09_test.location

Tabulka 5.1: *Seznam možných klíčových slov a k nim přiřazovaných názvů entitních množin*

prvek analyzován pomocí *NER* systému (více v podkapitole 4.1) a v závislosti na určené kategorii prvku může být do seznamu hlavních tabulek vložen další záznam.

V momentě, kdy je dokončena analýza vstupních seznamů, se dále zjišťuje, zda nebyl zadán požadavek na vypsání všech dostupných informací o daném člověku (v tom případě je třeba zpětně vložit do slovníku `mainTables` všechny užitečné entitní množiny) nebo na zjištění údajů ohledně konkrétního projektu (bude nutné položit DB více dotazů).

Po přidání všech hlavních tabulek do seznamu je prostřednictvím metody `addJoinTables` naplněn další seznam, jenž v projektu prezentuje spojovací tabulky a k nim příslušející SQL kód. Tyto spojovací tabulky jsou přidávány na základě výskytu hlavních entitních množin - vždy je potřeba dvou hlavních tabulek, které je mezi sebou nutno propojit. Z tohoto popisu lze tedy vyvodit celkový koncept tvorby dotazu, kde je nejprve zjištěno, jaké hlavní entitní množiny budou v dotazu figurovat a poté je z těchto množin odvozena přítomnost spojovacích tabulek.

Samotná tvorba SQL dotazu je prováděno v metodě `createQuery`, v níž je logicky spojen obsah slovníku s hlavními tabulkami a seznamu s tabulkami spojovacími. V relaci s tímto úkonem je důležité správným způsobem vytvořit klauzuli WHERE, neboť mohou nastat následující situace:

- klauzule WHERE není přítomna
- klauzule WHERE je vytvořena z uživatelem zadaného slova
- klauzule WHERE je vytvořena s pomocí seznamu nejcitovanějších autorů

V prvním případě není metoda pro tvorbu této klauzule vůbec volána, v dalších jsou volány odlišné metody. Společným jmenovatelem je však využití tzv. *stemmingu*(viz 4.2), starajícího se o konečný tvar klíčového slova. Takový tvar je pak databázi položen jako regulární výraz (znak "* " místo klasického rovnítka). Konkrétní tvary SQL dotazů lze vidět v podkapitole 3.2.3.

Posledním úkonem, který je pak prostřednictvím třídy `qaSqlCreator` proveden, je položení takto získaného dotazu databázi, čehož je docíleno pomocí již zmíněného modulu *PsycoPg* (3.3.1). V případě, že je v dotazu využívána entitní množina *keyword*, je třeba vytvořit dotaz několikrát, což již je v režii řídící funkce celého programu.

5.2.3 Třída qaPostProcess

Vstup této třídy tvoří data z databáze získaná prostřednictvím předchozího modulu `qaSqlCreator` a její primární funkcí je přetvořit tato data do přijatelného uživatelského výstupu.

Konkrétní způsob činnosti třídy `qaPostProcess` závisí na tom, zda je celý program spuštěn s vypínačem `-f` - poté jsou totiž přijatá data zpracována do uživatelsky přívětivé polohy. V opačném případě dostává uživatel na výstup seznamy v tomto tvaru:

```
[názvy získaných entitních množin][data pro každou entitní množinu]
```

Je patrné, že takový text není pro běžného uživatele vhodný pro čtení. Jedná se totiž o "surová data", získaná přímo z DB. Důvodem zachování takového výstupu v programu je využití ve spolupráci s GUI, o němž je pojednáváno v podkapitole 5.2.4.

Pro využití vyhledávače jako *stand-alone* programu je doporučeno použít přepínač `-f`, díky němuž jsou výše zmíněné seznamy zpracovány a uživateli poskytnuty ve formě přehledné ASCII tabulky. Hlavní řídící proces vykreslování této tabulky, představuje metoda `printFormattedOutput`, z níž je volána další metoda `drawTable`, starající se o tvorbu jednotlivých částí výsledného výstupu. Z důvodu často velkého rozsahu tabulky do šířky je doporučeno použít u volitelného parametru `-f` i název souboru, do něhož potom budou výsledky vytištěny. Zjednodušená podoba zformátovaného výstupu je zobrazena na obrázku 5.2.

```
+-----------+-------------+-------+--------------------------+
|Full Name  |Publication  |Country|Name of file in the RRS DB |
+-----------+-------------+-------+--------------------------+
|Jméno1     |Publikace1   |Země1  |Název souboru1             |
|Jméno2     |Publikace2   |Země2  |Název souboru2             |
+-----------+-------------+-------+--------------------------+
```

Obrázek 5.2: *Ilustrační obrázek zobrazující výslednou výstupní tabulku*

5.2.4 Webové GUI

Samotné GUI je součástí již rozpracovaného prostředí projektu *ReResearch* a nachází se na odkazu `http://athena2.fit.vutbr.cz:30101/rrsgui_devel/xmicul03/www/`. Pro započetí vyhledávání pak stačí z rolovacího menu zvolit možnost *All* a poté zadat svůj dotaz.

Protože je GUI vypracováno ve frameworku *Nette*[1], tvoří implementaci dva soubory - presenter `AllPresenter.php`, v němž se také nachází hlavní zdrojový kód GUI, a šablona `Search.phtml`, kde je realizováno grafické rozhraní.

Celá spolupráce PHP a python skriptu funguje tak, že prostřednictvím PHP funkce `proc_open()` je spuštěn skript qaMain.py, tomu je předána otázka z vyhledávací řádky, a následně je z něj získán výstup v podobě seznamu entitních množin a jednotlivých řádků tabulky (data jsou v neformátovaném tvaru, viz podkapitola 5.2.3). Tento výstup je potom v režii PHP kódu pomocí regulárních výrazů zpracován a výsledkem je jedna či více (je-li to třeba) tabulek. Celkove vzezření lze vidět na obrázku 5.3

[1] http://nette.org/cs/

Obrázek 5.3: *GUI projektu s výsledky na dotaz "Publications from french authors"*

Kapitola 6

Vyhodnocení systému

V momentě, kdy je cílový produkt dokončen, je nutné určitým způsobem vyhodnotit, zda výsledek práce vyhovuje požadavkům, vzneseným před samotným začátkem tvorby. O samotném procesu vyhodnocení pojednává tato kapitola.

6.1 Přesnost a pokrytí

Před samotným začátkem testování projektu je nutné seznámit se se standardními vyhodnocovacími metrikami, které byly v průběhu vyhodnocování použity. Jedná se o metriky *přesnost* (angl. *precision*) a *pokrytí* (*recall*).

V souvislosti s později obdrženými výsledky projektu *QuestionAnswering* je důležité předem uvést pojmy, s nimiž se dále setkáme později v rámci této kapitoly. Pro matematické vyjádření přesnosti a pokrytí se počítá s tím, že každý analyzovaný případ může nabýt jedno z následujících označení:

pravdivě pozitivní Dále PP, v angličtině *true positive* označuje, že daný případ byl v určitém kontextu pozitivní, přičemž jeho pozitivita byla předem předpokládána.

falešně pozitivní Dále FP, v angličtině *false positive* označuje, že daný případ v určitém kontextu vyhodnocen jako negativní, ale předem byl předpokládán jako pozitivní

pravdivě negativní Dále PN, v angličtině *true negative* označuje, že případ byl v určitém kontextu negativní a jeho negativita byla předem předpokládána.

falešně negativní Dále FN, v angličtině *false negativee* označuje, že případ byl v určitém kontextu pozitivní, ale předem byl předpokládán jako negativní

Přesnost

Přesnost (angl. *precision*) označuje poměr předpokládaných relevantních nálezů získaných jako odpověď na dotaz, vůči stejně získaným skutečně relevantním nálezům. V oblasti *strojového učení* či *vyhledávání informací* se jedná o hlavní používanou metriku [15].

Ve vztahu k heslům uvedeným na začátku kapitoly 6.1 lze přesnost vyjádřit jako (dle [13]):

$$přesnost = \frac{PP}{PP + FP} \tag{6.1}$$

Pokud se vrátíme ke slovnímu popisu uvedeném na začátku této sekce, je nutné blíže definovat pojem *relevantní nález*. Ten označuje událost, která je v daném kontextu brána jako žádoucí - má-li např. algoritmus za úkol hledat ve větách předložky, pak se relevantním nálezem myslí každý korektní případ jejího nalezení.

Předpokládaným relevantním nálezem v rámci tohoto projektu se pak rozumí všechny publikace, které jsou zobrazeny jako výsledek, skutečně relevantní nález lze následně určit pouze po manuálním přezkoumání každé z těchto zobrazených publikací.

Pokrytí

Pokrytí, v angličtině *recall*, zobrazuje poměr mezi všemi skutečně relevantními nálezy, zahrnutými v dotazu, a reálně relevantními nálezy, získanými v rámci odpovědi na dotaz. V oblasti *vyhledávání informací* není na *pokrytí* kladen takový důraz jako na *přesnost*, což bývá argumentováno faktem, že v případě hledání relevantních dokumentů vůči daným informacím je vždy nalezena jen určitá podmnožina skutečně relevantních článků a o reálné relevanci dokumentů, které nebyly získány, nemůžeme ve skutečnosti nic vědět [15].

Pro doplnění se opět zaměřme na pojmy uvedené na začátku kapitoly 6.1, prostřednictvím kterých je pokrytí vyjádřeno následovně (dle [13]):

$$pokrytí = \frac{PP}{PP + PN} \tag{6.2}$$

6.2 Způsob vyhodnocení

V rámci co nejrelevantnějšího určení výše popsaných metrik byly do úvahy vzaty dvě hlediska, ze kterých lze vypracovaný systém hodnotit:

- hodnocení z pohledu relevance výstupního textu vůči zadanému dotazu
- hodnocení z pohledu relevance výstupního textu vůči reálným datům

V případě první možnosti zde hlavní problém představuje proces samotného ohodnocování takto získaných dat. Pokud bychom data hodnotili pouze na základě jejich přečtení bez jakýchkoliv souvislostí, nejednalo by se o příliš relevantní metodiku a celé testování by bylo ovlivněno subjektivními pohledy. Případné ověřování relevance dat na internetu by mohlo přinést určité výsledky, za prvé však jde o relativně pracnou metodiku, neboť ne všechny informace je možno nalézt pouhým zadáním hesla do vyhledávače a za druhé by zde stále přetrvával problém neuspokojivého určení hodnoty *pokrytí*.

Tento problém již byl popsán v podkapitole 6.1 a spočívá v nemožnosti zvolit vhodnou hodnotu jmenovatele do vzorce pro výpočet *pokrytí*. Pokud bychom zvolili nejjednodušší cestu (předpokládejme, že ruční procházení položek DB a hledání všech relevantních dat, která mají být vypsána, není kvůli obrovské časové náročnosti žádoucí) a za *skutečně relevantní data* pokládali všechna data získaná v rámci dotazu, nabývala by v konečném důsledku tato metrika hodnoty 100%. Takový výsledek by se za relevantní dal pokládat jen stěží.

V konečném důsledku je tedy zvolena druhá z výše zmíněných možností, kdy je předem předpokládáno, že získané výsledky jsou vzhledem k uživatelskému dotazu relevantní[1], avšak abychom samotnou relevanci potvrdili, a vyvrátili tak podezření např. o nesmyslných

[1] jde o opodstatněný předpoklad - SQL dotaz položený DB byl ve všech testovaných případech v takovém tvaru, v jakém byl očekáván

názvech publikací či projektů, je nutné ručně zkontrolovat obsah zobrazených dat, resp. publikací (je-li to možné) a na základě tohoto faktu pak *přesnost* a *pokrytí* vypočítat.

Aby bylo možné tento úkon provést, je třeba, aby získaná data splňovaly dvě podmínky - ve výstupním textu musí být přítomny názvy publikací a zároveň se některé z těchto publikací musí nacházet v datovém skladu projektu *ReResearch* (výpis případných jmen souborů ve skladu lze získat prostřednictvím vypínače `-t`). Poté je možné začít porovnávat názvy publikací a jména autorů se skutečnými publikacemi a na základě jejich shody určit *přesnost*. Co se týká *pokrytí*, bylo rozhodnuto, že určováno nebude z důvodu neúměrných časových nároků, potřebných k případné ruční kontrole celé DB. V souvislosti s určením *přesnosti* je ještě třeba uvést, že pokud je u některých dotazů zobrazen pouze jejich název a jméno autora v DB chybí, je validita určena pouze podle dostupného názvu bez ohledu na jméno.

Je však nutné říci, že ani přes zvolení takového způsobu testování nelze hovořit o bezchybné metodice, jelikož se zde z větší části namísto hodnocení skutečných kvalit výsledného projektu jedná o vyhodnocování samotných dat, dostupných v DB.

6.3 Bližší pohled na vybrané testy

V této podkapitole bude hlavní důraz kladen na popis některých zajímavých částí z procesu vyhodnocování systému. V případě zájmu o hodnoty metrik dotazů, uvedených pod pojmem "Další variace dotazu" lze nahlédnout do kompletní tabulky se všemi testovými otázkami, která je součástí přílohy.

Základní filozofie testování spočívá ve vícenásobném položení dotazu téže podstaty, ale pokaždé v jiném tvaru, čímž lze demonstrovat schopnost programu zpracovávat více možností zadání - je podporováno jak zadání v přirozeném jazyce (angličtina), tak ve formě hesel.

Vyhledání informací o zadané osobě
Položený dotaz: *give me all available information about David Johnson*
Zahrnutých publikací v datovém skladu: 4
Relevantní publikace: 1
Přesnost: 25%

Dotaz na konkrétního výzkumníka by měl přinést data z několika důležitých tabulek (viz 3.2.3), přičemž na uživateli je ponecháno rozhodnutí, zda do vyhledávacího pole zadá celé jméno, či např. jen iniciály křestního jména (v tomto případě *D. Johnson*). Jelikož bylo rozhodnuto, že do uživatelského vstupu bude zasahováno jen minimálně, je důležité zmínit, že obdržené výsledky se budou v závislosti na těchto zadáních lišit, protože stejně tak se liší i záznamy jmen v DB.

Jeden z prvních provedených testů bohužel hned ze začátku poukazuje na horší kvalitu dat v databázi. Ze čtyř publikací, které se v rámci dotazu vyskytovaly v datovém skladu, byla relevantní pouze jedna. Problémem je fakt, že i když názvy zobrazených publikací souhlasí s kontrolovanými články, nemohou být takto získaná data označena za relevantní, neboť dotazovaný David Johnson je autorem pouze jednoho ze zkontrolovaných děl. Konkrétní příčina tohoto jevu bude nejspíše spočívat v nesprávných vazbách v tabulce *j_pers_publ*.

Další variace dotazu: *David Johnson; give me all known info about David Johnson; get list of all articles, which was written by David Johnson* atd.

Vyhledání informací o osobách z daných lokací
Položený dotaz: *researchers from europe and their publications*
Zahrnutých publikací v datovém skladu: 33
Relevantní publikace: 23
Přesnost: 69,7%

Ze všech takto získaných a ověřovaných publikací odpovídaly obdrženému popisu celé dvě třetiny, a tak se z pohledu relevance dá hovořit o relativně uspokojivém číslu. Problémem je malé množství nalezených položek, což u těchto typů dotazů způsobují chybějící záznamy v potřebných spojovacích tabulkách. Pro ověření tohoto faktu stačí, aby v dotazu nebylo specifikováno klíčové slovo *publications* a počet položek se dostane na číslo 26457, což dostatečně nenaplněnost *join* tabulek ilustruje.

Při kontrole těchto publikací jsem navíc narazil na určité nesrovnalosti ve skutečných jménech autorů a položkách databáze, z nichž byly často odstraněny jejich příjmení, a tak jsou některé výpisy nekorektní. Jako příklad uveďme jména *James M. Buchanan*, *James M. Hyman* nebo *James M. Tiedje*, která jsou v DB uložena jako *M. James*.

Další variace dotazu a podobné dotazy:*list all french, italian, german and czech researchers; Give me list of authors from brno university; give me info about publications from french, italian, german and czech researchers; get all suitable info about publications written by authors from european countries*

Informace o nejcitovanějších autorech
Příklad položeného dotazu: *get the most cited authors from united kingdom and their publications*
Zahrnutých publikací v datovém skladu: 19
Relevantní publikace: 11
Přesnost: 57,9%

U tohoto dotazu se poprvé výrazněji projevuje nedokonalost automatické extrakce důležitých informací z článků. Všechny nevalidní nálezy jsou zde totiž zapříčiněny špatným názvem publikace, za něhož je vydávána organizace, pod jejíž záštitou daný článek vychází. Dostane-li uživatel na výstupu taková data, nelze je samozřejmě považovat za relevantní, neboť se nedá hovořit o jejich informačním přínosu.

Tento problém však nebyl jediný. Z neznámého důvodu se u takto postižených publikací nepodařilo správně určit ani autora, který k nim měl dle výsledků příslušet, a tak chyba nebyla pouze u špatné extrakce dat, ale nejspíše i v dezinformacích v tabulce j_file_publ.

Další variace dotazu a podobné dotazy: *list 200 most cited authors and their publications; give me list of most cited authors; give me 100 most cited authors*

Informace o zadaném tématu
Příklad položeného dotazu: *get all suitable info about education*
Zahrnutých publikací v datovém skladu: 880 (pro test vybráno 20 náhodných)
Relevantní publikace: 16
Přesnost: 80%

U těchto typů dotazů není z důvodu velkého počtu získaných položek možno určit *pokrytí*, protože z celkového počtu 880 publikací bylo náhodně vybráno 20 a ty byly zkoumány. Počítat *pokrytí* ze všech publikací tak není relevantní.

Přesnost 80% se jeví jako úspěšné číslo, je však důležité vědět, že tohoto výsledku bylo dosaženo kontrolou pouhých názvů publikací, jelikož jména autorů nebyly v DB dostupné.

Opět se zde ve velké míře projevuje problém se špatnou extrakcí dat z článků, kdy je místo názvu publikace uveden název organizace. Nejhorších výsledků z celého testování bylo dosaženo při zadání dotazu "*get all suitable info about education*", přičemž zde *přesnost* představovala hodnotu pouhých 2/20.

Svůj podíl na tomto problému měla jednak špatná extrakce informací, hlavní příčinou se však ukázalo být obrovské množství nevalidních dat ve spojovací tabulce *j_file_publ*, která má za úkol propojovat publikace se jmény souborů z datového úložiště systému *RRS*. Tento problém pokládám za obzvlášť závažný, neboť kromě obecně nevalidních informací je v tabulce *j_file_publ* také mnoho duplicitních dat, která mimo jiné zapříčiňují mnohem větší množství vypsaných položek při aktivním vypínači `-t`[2], než při výpisu dat bez tohoto parametru. U již zmíněného dotazu "*get all suitable info about education*" je při spuštění s tímto parametrem zobrazeno více než 81000 položek, zatímco bez něj jde o číslo nepřekračující 3000. Jistou měrou k tomu přispívá špatný tvar slova *education* v klauzuli WHERE, který je prostřednictvím *Porter stemmeru* zkrácen na "*educ*", přesto je tento případ pro ilustraci závažnosti problému dostačující.

Jediným ospravedlněním této problematiky může být fakt, že při využití z GUI se názvy souborů nevypisují, a tak jsou takto získané informace relativně uspokojivé.

Další variace dotazu a podobné dotazy: *education; list all publications about education*

6.4 Zhodnocení celkových výsledků

Kompletní výsledky provedeného vyhodnocení jsou k dispozici v příloze. Bohužel se více než o hodnocení skutečných kvalit systému jedná o hodnocení kvality dat v databázi, s čímž ovšem bylo počítáno již od samého začátku. Výsledkem provedeného vyhodnocení jsou tedy tyto zásadní nedostatky zjištěné v databázi systému *Reresearch*:

- špatný tvar jména v DB (James M. Buchanan -> M. James)
- místo názvu publikací uvedeny názvy organizací
- opravdu velké množství duplicitních vazeb v tabulce *j_file_publ*
- obecně nevalidní data v tabulce *j_file_publ*
- velká absence důležitých dat v tabulce *j_pers_publ*

[2] zajišťuje zobrazování názvů souborů z úložiště

Tyto skutečnosti se všechny určitou měrou podílí na výsledku, zobrazeném uživateli. Jako hlavní problém v současnosti shledávám skutečně malou množinu zobrazených informací při konkrétnějších požadavcích (např. zobrazení publikací francouzských autorů), což je způsobeno obecnou nenaplněností spojovacích tabulek.

Dále představuje velký problém mnoho duplicitních a nevalidních dat v tabulce *j_file_publ*, kdy na jednu publikaci připadá několik desítek autorů. Tento nedostatek se sice neprojeví při práci s GUI, ale přesto jde o důležitý poznatek, který není na místě přehlížet.

Kapitola 7

Srovnání s jinými vyhledávači

Kromě základního vyhodnocení vytvořeného systému na vlastních testech v kapitole 6 je jedním z bodů posuzování kvality konečného produktu také srovnání s jinými, již existujícími implementacemi vyhledávačů. Problematice srovnávání dostupných vyhledávacích systémů je věnována tato kapitola.

7.1 Způsob srovnávání vyhledávačů

Budeme-li se zabývat porovnáváním různých produktů s podobným účelem využití, je vždy na místě ptát se, z jakého hlediska bude srovnání provedeno tak, aby vydedukované výsledky byly co nejvíce relevantní a měly vypovídající hodnotu. Jinými slovy je potřeba určit oblasti, které mohou mít porovnávané produkty společné a následně v mezích těchto kategorií provést relevantní srovnání.

V případě vypracovaného systému *QuestionAnswering* by bylo ideální nalézt a srovnat vyhledávače, které:

- pracují nad databází s vědeckými daty
- nabízejí inteligentní vyhledávání

Bohužel, v současné době není možné narazit na vědecký vyhledávací systém, jenž by se svým konceptem zaměřoval na inteligentní vyhledávání dotazů. Z toho důvodu je nutné vybírat testované kategorie s ohledem na přijatelnost pro všechny porovnávané produkty, protože je však primárním účelem srovnat a zhodnotit vytvořený systém, bude se celé testování odvíjet od sady dotazů, které byly použity i k samotnému testování. Každý dotaz však bude upraven do takové podoby, u níž se dá počítat s možností zadání uživatelem i u ostatních vyhledávačů.

Do srovnání byly zařazeny dva vědecké portály nabízející svůj vlastní vyhledávací systém - Google Scholar[1] a Microsoft Adademic Search[2]. Nutno ještě dodat, že výsledky srovnání budou ovlivněny faktem, známým již z kapitoly 6 - tím je nízká kvalita dat v DB systému *ReResearch*.

[1] http://scholar.google.cz/
[2] http://academic.research.microsoft.com/

7.2 Srovnání podle jednotlivých typů dotazů

Následuje přehled a srovnání odpovědí na zadané dotazy. Každý získaný výsledek je porovnáván z hlediska dvou kritérií - v jaké kvalitě byla nalezena požadovaná informace a jaké množství takových informací bylo. I když je v následných komentářích vždy přítomna snaha zachovat maximální objektivitu, vzhledem k rozdílnosti srovnávaných systémů je třeba přihlížet k faktu, že na daný problém nelze aplikovat žádnou ze standardních metrik, a tak není možné brát tyto komentáře jako podložená fakta - vždy je zde promítnuta určitá míra subjektivity. Jako prostředek pro vytvoření si přehledu o kvalitě vyhotoveného vyhledávače, resp. kvalitu dat v DB, je však takové srovnání vyhovující.

Vyhledání informací o zadané osobě
Položený dotaz: *David Johnson*

Jako ideální odpověď na takový dotaz se jeví odkaz přímo na profil daného výzkumníka, případně výpis jeho publikací. Jelikož je zobrazování autorských profilů již mimo rozsah vypracovaného systému *QuestionAnswering*, lze od tohoto vyhledávače očekávat pouze zobrazení publikací a doplňujících informací jako projekt, e-mail a země původu. Bohužel žádná z takových informací není v tomto konkrétním výpisu přítomna, a tak veškerá získaná data tvoří pouze sedm publikací (což je problém formátů jmen v DB, viz kap. 6).

To je oproti zbylým vyhledávačům neuspokojivý výsledek, neboť ty nabízejí jednoduchý přístup k autorským profilům a velké množství nalezených informací (*MSAS* udává 34312 publikací, *Google Scholar* pak 2880000 celkově nalezených výsledků) týkající se autorů daného jména.

Záměrně je zde uvedeno slovo autor v množném čísle, objevuje se zde totiž souvislost s problémem mnohoznačnosti dat v DB *ReResearch*, kdy je po zadání dotazu *David Johnson* uživateli zobrazeno několik publikací od Davida Johnsona, bohužel již ale nelze zjistit, že ne vždy se jedná o jednoho autora všech děl. Tento fakt souvisí opět s nekvalitními daty v DB a u zbylých dvou vyhledávačů nepředstavuje problém takového rozsahu, hodného ke zmínění.

Vyhledání informací ohledně účasti na projektu
Položený dotaz: *projects of Nathan Paxton*

Cílem při zadání této věty by mělo být co nejrychlejší vyhledání informací ohledně projektů, s nimiž je zadaná osoba ve spojitosti. I když na uvedené zadání nalezl systém *QuestionAnswering* pouze jeden vyhovující záznam, stále se dá mluvit o zajímavějším výsledku, než v případě obou zbývajících vyhledávačů, kde se v lepším případě (*Google Scholar*) podařilo nalézt pouze několik málo publikací od tohoto autora.

Tento fakt je samozřejmě pochopitelný, neboť u těchto systémů není prováděno žádné zpracování vstupu. Úspěchu však nebylo dosaženo ani v případě, že bylo zadáno pouze samotné jméno autora a jeho účast na projektech pak vyhledávána cíleně skrze autorský profil či publikace. Zde se tedy objevuje první výhoda systému *QuestionAnswering* - za předpokladu naplněnosti všech potřebných tabulek DB je možno použít jej pro rychlé zjištění příslušnosti autora k určitému projektu.

Informace o autorech z dané země
Položený dotaz: *researchers from France*

Budeme-li předpokládat, že odpovědí na tento dotaz má být seznam francouzských autorů, pak nejrelevantnější výsledky opět přináší vyhledávač *QuestionAnswering*. Ostatní vyhledávače se pokoušejí nalézat řešení v názvech publikací, což logicky nevede k požadované informační hodnotě.

Při pokusu nalézt ve srovnávaných vyhledávačích filtr, který by umožňoval vyhledávání autorů dle národnosti, skončil neúspěchem, a tak se zde projevuje další užitečná funkce vypracovaného systému.

Informace o autorech ze zadané univerzity
Položený dotaz: *authors from brno university*

Zde pro jeden vyhledávač nastává podobný problém, jako v případě předchozí otázky. Jelikož je tento dotaz v režii *Google Scholar* opět porovnáván s názvy, případně abstrakty vědeckých prací, nelze čekat přímo seznam všech výzkumníků z brněnských univerzit. I přesto mohou být odkazy na takto získané publikace užitečné, neboť se zde zmínka o brněnských univerzitách často nachází v souvislosti s autory dané práce.

MS Academic Search nabízí přímo vyhledávání dle názvu univerzity, na dotaz *brno university* však není nalezen žádný relevantní záznam. Po zadání řetězce *harvard university* je však uživatel přesměrován na profil této instituce a ihned mu je k dispozici seznam, čítající dle statistik 42430 autorů a 598480 publikací spjatých s harvardskou universitou.

Systém *QuestionAnswering* uživateli nabízí jako odpověď na původní dotaz seznam, čítající čtyři autory z VUT Brno bez jakýchkoliv doplňujících informací. Na zadání *authors from harvard university* pak nevrací žádný výsledek, což lze opět odůvodnit špatnými či chybějícími daty v DB.

Seznamy nejcitovanějších autorů
Položený dotaz: *most cited authors*

Problematika citovaných autorů ve vztahu k vypracovávanému projektu byla již osvětlena v sekci 3.2.4. Jako odpověď na daný dotaz je vypsán obsah seznamu[3] ze serveru *CiteSeerX*, který může posloužit jako rychlý ukazatel zájmu o dané výzkumníky, avšak jeho největším nedostatkem je fakt, že slučuje stejná jména do jedné položky. I přes tuto skutečnost se ale jedná o obsáhlý seznam, k němuž se skrz zbylé dva vyhledávače nelze dostat.

Jejich devízou je spíše zobrazení nejcitovanějších autorů z určité oblasti výzkumu, což je opět dáno zaměřením výsledků vyhledávání na konkrétní publikace, které se těmito srovnáními zabývají. Získané seznamy citovanosti tak nejsou přímým výstupem srovnávaných vyhledávačů, ale obsahem nalezených publikací.

Na samotném uživateli pak závisí, jakou z variant zvolí - zda stručný, ale všeobecný a obsáhlý přehled, jaký nabízí systém *QuestionAnswering*, nebo podrobnější výsledky z určité oblasti zprostředkované skrze tématicky zaměřené publikace.

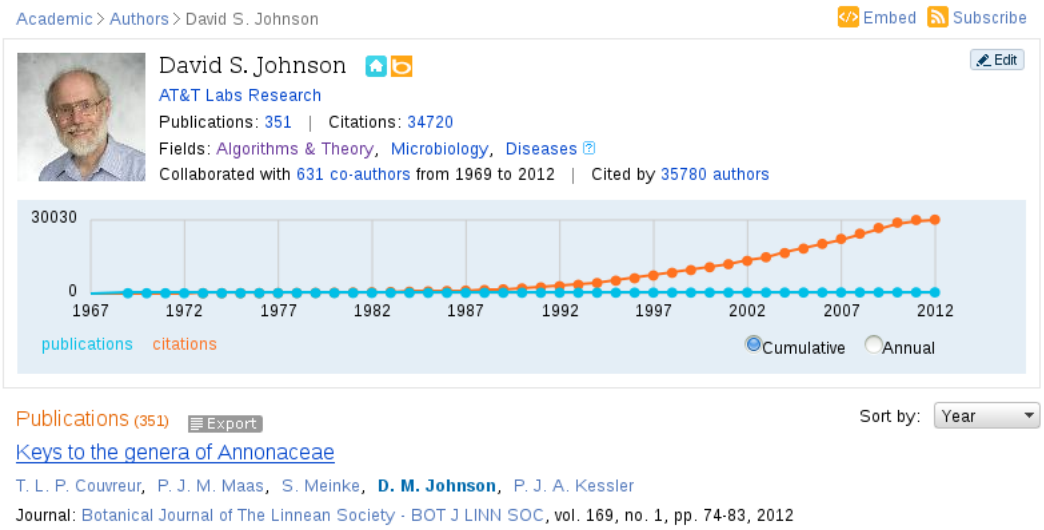
[3] http://citeseerx.ist.psu.edu/stats/authors?all=true

Vyhledání informací o daném tématu
Položený dotaz: *education*

U zavedených vyhledávacích systémů se dá opět předpokládat, že jejich hlavní odpovědí na tyto obecné typy dotazů budou publikace s daným klíčovým slovem v názvu. Tento způsob zadávání otázek patří mezi nejčastější formy hledání informací, takže není žádným překvapením, že se předešlý předpoklad v praxi naplňuje.

Vyhledávač *QuestionAnswering* byl vyvíjen s cílem nabídnout data s adekvátní informační hodnotou vůči jiným systémům. Při zadání dotazu *education* je výsledek rozdělen do dvou částí - projekty odpovídající heslu "education" a publikace odpovídající heslu "education". Informace ohledně publikací bohužel z velké části nejsou kompletní díky chybějícím datům v DB, na druhou stranu se zde názorně dá poukázat na přínos využití tabulky *keyword*.

Oproti srovnávaným vyhledávačům jsou zde totiž zohledněny i výsledky, které nemají zadané klíčové slovo přímo v názvu či v abstraktu, a tak může být množina získaných dat mnohem variabilnější a nemusí se týkat jen děl, která mají dané téma explicitně určeno.



Obrázek 7.1: *Příklad autorského profilu na stránkách MS Academic Search*

7.3 Výsledné zhodnocení provedeného srovnání

Z provedeného srovnání vyplynulo, že vyhledávač *QuestionAnswering* se osvědčí zejména jako prostředek k rychlému získání obecných informací. Uživatel pouze konkrétně specifikuje svůj dotaz a pokud jsou k dané oblasti dispozici data, během několika okamžiků jsou mu k dispozici. S takto jednoduchým přístupem bohužel další srovnávané vyhledávače nepočítají a případné informace z konkrétních oblastí je nutno vyhledávat pracnějším způsobem.

Problémem dat získaných od vyhotoveného systému je jejich neinteraktivita, a tím pádem i nízká informační hodnota (což není chybou samotného vyhledávače - interaktivita a provázanost s ostatními daty je již v režii dalších částí informačního systému projektu

ReResearch, jejichž tvorba není předmětem této práce), která tak činí systém vhodným pro vyhledání obecných dat, pro jejichž reálný průzkum již je doporučeno poohlédnout se jinam. Dalším problémem je již zmiňovaná nedostatečnost dat v databázi, díky níž zůstává mnoho důležitých informací skryto.

Do budoucna bude jistě dobrou volbou nechat se inspirovat současným stavem profilů výzkumníků (7.1) a jejich publikací ve vyhledávacích systémech *Google Scholar* a *MS Academic Search*, což je oblast, v níž v rámci provedeného srovnání systémy skutečně dominovaly a množství informací, nalezených po zadání konkrétního jména, bylo vždy relevantnější, než v případě systému *QuestionAnswering*.

Kapitola 8

Závěr

V rámci této práce byl vytvořen systém *QuestionAnswering*, který slouží jako vyhledávač pracující nad databází projektu *ReResearch*. Tento vyhledávač se svým konceptem dotýká oblasti *odpovídání na otázky*, neboť umožňuje uživateli zadávat dotazy ve formě přirozeného jazyka (angličtina). V rámci snahy o přiblížení se již existujícím variantám databázových vyhledávačů jsou však podporovány také dotazy sestávající pouze z jednotlivých klíčových slov.

Vytvořený systém byl vyhodnocen dle standardních metrik nazvaných *přesnost* a *pokrytí*, z celé kapitoly vyhodnocení systému však bohužel vyplynulo, že se databáze projektu *ReResearch* v současnosti potýká s nízkou kvalitou obsažených dat. Zejména spojovací tabulky trpí nedostatkem relevantních informací, což má za následek menší počet systémem nalezených odpovědí na zadané otázky konkrétnějšího charakteru.

Přesto se však systém *QuestionAnswering* osvědčí jako prostředek k rychlému získání obecných informací, jak se lze dozvědět z provedeného srovnání se systémy *Google Scholar* a *Microsoft Academic Search*. Oproti těmto systémům nabízí vyhledávač *QuestionAnswering* menší počet nalezených položek a nemůže se s nimi rovnat co do množství poskytovaných informací o daných vědeckých subjektech, avšak v případě, kdy uživatel potřebuje co nejrychleji zjistit specifičtěji zaměřená fakta, jako např. příslušnost autorů k dané národnosti či obecnou citovanost autorů, dokáže vypracovaný systém poskytnout základní, avšak dostatečně relevantní informace. Nejen díky tomuto faktu se dá konstatovat, že cíle práce se podařilo splnit.

Literatura

[1] Alfonseca, E.; Manandhar, S.: An Unsupervised Method for General Named Entity Recognition And Automated Concept Discovery. In *In: Proceedings of the 1 st International Conference on General WordNet*, 2002.

[2] Bikel, D. M.; Miller, S.; Schwartz, R.; aj.: Nymble: a High-Performance Learning Name-finder. In *In Proceedings of the Fifth Conference on Applied Natural Language Processing*, 1997, s. 194–201.

[3] Djahantighi, F.; Norouzifard, M.; Davarpanah, S.; aj.: Using Natural Language Processing in Order to Create SQL Queries. *Proceedings of the International Conference on Computer and Communication Engineering 2008*, May 2008.

[4] Heller, S.: ReReSearch: Databáze - popis schématu [online]. https://merlin.fit.vutbr.cz/nlp-wiki/index.php/Rrs_db:Schema, [cit. 2013-04-25], interní dokumentace.

[5] Heller, S.: Knihovna pro Podporu Vývoje Systému ReReSearch, bakalářská práce. 2011, fIT VUT v Brně.

[6] Hirschman, L.; Gaizauskas, R.: Natural Language Question Answering: The View from Here. *Nat. Lang. Eng.*, 2001: s. 275–300, ISSN 1351-3249.

[7] Jawadekar, W.: *Knowledge Management: Text & Cases*. McGraw-Hill Education (India) Pvt Limited, 2011, ISBN 9780070700864.

[8] Jurafsky, D.; Martin, J.: *Speech And Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall series in artificial intelligence, Pearson Prentice Hall/Pearson education international, 2009, ISBN 9780131873216.

[9] Kaur, S.; Bali, R. S.: Sql Generation And Execution From Natural Language Processing. *International Journal of Computing & Business Research*, 2012.

[10] Lampert, A.: A Quick Introduction to Question Answering. 2004.

[11] McCallum, A.; Li, W.: Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, Association for Computational Linguistics, 2003, s. 188–191.

[12] Nadeau, D.; Sekine, S.: A Survey of Named Entity Recognition and Classification. *Linguisticae Investigationes*, January 2007, publisher: John Benjamins Publishing Company.

[13] Olson, D.; Delen, D.: *Advanced Data Mining Techniques*. Springer-Verlag Berlin Heidelberg, 2008, ISBN 9783540769170.

[14] Porter, M.: The Porter Stemming Algorithm. http://tartarus.org/martin/PorterStemmer/index.html, 2006, [cit. 2013-05-7].

[15] Powers, D. M.: Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies 2011*: s. 37–63.

[16] van Rijsbergen, C.; Robertson, S.; Porter, M.: New Models in Probabilistic Information Retrieval. 1980.

[17] Riloff, E.; Jones, R.: Learning Dictionaries for Information Extraction by Multi-level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference*, American Association for Artificial Intelligence, 1999, s. 474–479.

[18] Sharma, D.: Article: Stemming Algorithms: A Comparative Study and their Analysis. *International Journal of Applied Information Systems*, September 2012: s. 7–12, published by Foundation of Computer Science, New York, USA.

Příloha A

Obsah DVD

Seznam jednotlivých složek:

bp

Složka obsahující elektronickou verzi této práce ve formátu PDF.

- source obsahuje zdrojové LaTeXové soubory této práce

rrs_qa2

Složka obsahující data týkající se vypracovaného systému *QuestionAnswering*

- bin obsahuje zdrojové **.py* soubory a složku *ner*, v níž se nachází *Stanford NER interface for Python*

- stanford-ner-2012-11-11 verze *Stanford NER*, pod nímž byl tento program testován - obsahuje i využívané modelové třídy

- test zde se nachází číselně pojmenované složky s výsledky jednotlivých textů a také samotný testovací skript

Příloha B

Manuál

B.1 Přístup skrze webové GUI

Adresa: http://athena2.fit.vutbr.cz:30101/rrsgui_devel/xmicul03/www/

Z rolovací nabídky poté vybrat možnost *All* a začít vyhledávat.

Jediným možným problémem, který může nastat, je náhlé vypnutí *NER* serveru. V tom případě GUI vypisuje chybovou hlášku a je potřeba *NER* opět zapnout (standardně běží na pozadí v rámci utility *screen*, je doporučeno zapínat jej v rámci této utility i nadále). Toho lze docílit pomocí příkazu:

```
java -mx1000m -cp stanford-ner.jar edu.stanford.nlp.ie.NERServer
-loadClassifier classifiers/
english.conll.4class.caseless.distsim.crf.ser.gz -port 8080
```

Příkaz je nutné zadat z umístění, na němž je stažen a rozbalen balík *Stanford NER*. V případě serveru athena se *Stanford NER* nachází zde:

```
sftp://athena3.fit.vutbr.cz/mnt/minerva1/nlp-in/athena3/reresearch/
public.www/root/rrsgui_devel/xmicul03/stanford-ner-2012-11-11
```

Po splnění těchto požadavků pak *NER* server běží na portu 8080 a systém *QuestionAnswering* je možno začít používat.

B.2 Spuštění jako stand-alone aplikace

Výchozím souborem je soubor `qaMain.py`, jež lze spustit s následujícími parametry:

```
python qaMain.py [-f [SOUBOR]][-t]
```

-f [SOUBOR] : Bez použití tohoto parametru je výstupem skriptu neformátovaný text, obsahující nejprve témata výsledných tabulek a poté řádky těchto tabulek. Tento text je primárně využit jako výstup pro GUI a pro běžné čtení výsledků se nehodí. Pokud používáme skript samostatně, je doporučeno přepínač použít, neboť pak jsou získaná data zformátována do tvaru přehledné ASCII tabulky. Kvůli pravděpodobně velkému rozpětí výsledné tabulky do šířky je doporučeno také využívat volitelného argumentu `SOUBOR`, který výstup vypíše do souboru zadaného jména.

-t : Volitelný parametr, který v případě, že získaná data zahrnují publikace, přidá do výsledného výstupu jména souborů, které dané publikace představují v rámci datového úložiště projektu *ReResearch*. Tento parametr byl využíván primárně pro testování.

B.2.1 Využívané utility a knihovny

Pro úspěšné využívání tohoto programu je nutné mít zprovozněny následující knihovny nebo utility:

Stanford Named Entity Recognizer [1] Nutná součást systému *QuestionAnswering*, vždy před jeho použitím je zapotřebí spustit *NER* server příkazem uvedeným v části přílohy B.1. Umístění *Stanford NER* na lokálním PC nehraje roli, důležitou prerekvizitu představuje pouze přidání souboru *stanford-ner.jar* do *java CLASSPATH*.

Python interface for Stanford NER [2] Rozhraní pro jazyk *Python*, umožňující využívat běžícího *NER* serveru. Je nutné mít složku s touto utilitou ve stejném umístění, jako jsou zdrojové soubory. Na přiloženém DVD jde o složku `rrs_qa2/bin/ner`

Porter stemmer algoritmus z knihovny NLTK [3] Rozšířená knihovna pro jazyk *Python*, z níž je využíván právě algoritmus *Porter stemmer*.

Knihovna PsycoPg2 [4] Knihovna použitá pro přístup k *PostgreSQL* databázi. Více v hlavní části práce.

[1] http://nlp.stanford.edu/software/CRF-NER.shtml#Download
[2] https://github.com/dat/pyner
[3] http://nltk.org/api/nltk.stem.html?highlight=porter%20stemmer#nltk.stem.porter.PorterStemmer
[4] http://initd.org/psycopg/docs/index.html

Příloha C

Tabulka provedených testů

Následuje tabulka, která představuje seznam provedených testů a jejich vyhodnocení. Testové otázky zároveň poukazují na schopnost programu zpracovávat různě formulované dotazy stejného typu. Právě proto je vždy jeden dotaz položen v několika variantách. Originální výstupy jednotlivých testů lze nalézt ve složkách `rrs_qa2/test/číslo_testu` na přiloženém DVD.

Zkratky uvedeny v pravé části tabulky znamenají následující:

- **PZP** - počet získaných položek
- **PJP** - počet jmen publikací (publikace, které se vyskytovaly v datovém skladu projektu *ReResearch* a byly ručně zkontrolovány)

Test	**Dotaz**	**Přesnost**	**PZP**	**PJP**
1	David Johnson	25%	7	4
2	give me all available information about David Johnson	25%	7	4
3	give me all known info about David Johnson	25%	7	4
4	David Johnson publications	25%	7	4
5	give me all publications from David Johnson	25%	7	4
6	list all publications, whose author is David Johnson	25%	7	4
7	get list of all articles, which was written by David Johnson	25%	7	4
8	Nathan Paxton projects	-	1	0
9	give me all projects, whose is participating Nathan Paxton	-	1	0
10	Nathan Paxton participation on projects	-	1	0
11	researchers from France	-	6686	0
12	list all french researchers	-	6686	0
13	list 20 french, italian, german and czech researchers	-	20	0
14	list all french, italian, german and czech researchers	-	11267	0
15	list all authors from france, italy, germany and czech republic	-	11267	0
16	list all french researchers and their publications	100%	4	2
17	list all authors from france, italy, germany and czech republic and their publications	100%	6	3
18	give me info about publications from french, italian, german and czech researchers	100%	6	3
19	give me list of authors from brno university	-	4	0
20	list of authors and their publications from umea university	100%	2	1
21	get list of european authors	-	26457	0

Test	Dotaz	Přesnost	PZP	PJP
22	give me information about authors from europe	-	26457	0
23	researchers from europe and their publications	69.7%	80	33
24	get all suitable info about authors from asian countries	-	362	0
25	get all suitable info about publications written by authors from european countries	69.7%	80	33
26	authors from american countries	-	56	0
27	give me list of most cited authors	-	10000	0
28	give me 100 most cited authors	-	100	0
29	get the most cited authors from united kingdom and their publications	57.9%	46	19
30	ten most cited authors from France	100%	2	1
31	give me five most cited authors from Ireland	-	5	0
32	list all projects about education	-	1955	0
33	give me all projects and publications about education	10%	81305	80801(20)
34	list 200 most cited authors and their publications	16.7%	201	54
35	list all publications about education	10%	81305	80801(20)
36	education	80%	2835	880(20)
37	get all suitable info about education	80%	2835	880(20)
38	blood projects	-	56	0